НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2020р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 121 Інженерія програмного забезпечення

на тему «Система збору даних про споживання енергоресурсів та температурні режими будівель для служби енергоменеджменту»

Виконав (-ла): студент (-ка) 4 курсу, групи ТВ-61

Бевза Дмитро Васильович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент Тарнавський Юрій Адамович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки: 121 Інженерія програмного забезпечення

Спеціалізація: Програмне забезпечення розподілених систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

___Бевзі Дмитру Васильовичу_____
(прізвище, ім’я, по батькові)

1. Тема роботи «Система збору даних про споживання енергоресурсів та_____ температурні режими будівель для служби енергоменеджменту»
_____

керівник роботи_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 202__р. № __

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи _____
_____

_____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Проаналізувати існуючі системи енергоменеджменту, дослідити можливість впровадження таких систем. Спроектувати та розробити систему для контролю температур та енергоресурсів для служби енергоменеджменту НТУУ «КПІ»_____
_____
_____
_____
_____
_____

5. Перелік ілюстративного матеріалу

_____
_____
_____
_____
_____

6. Дата видачі завдання ”___”_____201___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	10.11.2019	
2.	Вивчення та аналіз задачі	11.11.2019-20.12.2019	
3.	Розробка архітектури та загальної структури системи	07.01.2020-05.02.2020	
4.	Розробка структур окремих підсистем	06.02.2020-01.03.2020	
5.	Програмна реалізація системи	02.03.2020-10.05.2020	
6.	Оформлення пояснювальної записки	11.05.2020-17.05.2020	
7.	Захист програмного продукту	17.05.2020	
8.	Передзахист	12.05.2020	
9.	Захист	16.05.2020	

Студент _____ ___Бевза Д.В._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _Тарнавський Ю.А._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою роботи є створення системи для контролю енергоресурсів та температур у будівлях та аудиторіях.

Система допомагає автоматизувати процес аналізу температур у аудиторіях, формування звітів, зберігання, збір та обробку інформації про корпуси, аудиторії, тепло лічильники та температури.

Система передбачає наявність 4 ролей із різними можливостями доступу, що дозволяє розмежувати відповідальних осіб за наповнення чи використання системи згідно їх рівня доступу.

ABSTRACT

The purpose of work is to create system to control energy resources and temperatures in buildings and classrooms.

The system supports automatic technological processing of temperature in the auditorium, storing, storing, collecting and processing information about buildings, auditoriums, heat meters and temperatures.

The system provides 4 roles with different feed options that allow you to differentiate individuals for filling and apply the systems used.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – база даних

СКБД – система керування базами даних

CMS – (з англ. Content manadgement system) – система керування контентом

API – (з англ. Application programming interface) – прикладний програмний інтерфейс

ВСТУП

На сьогоднішній день проблема ефективного використання ресурсів є дуже важливою. У всьому світі звертають увагу на проблеми екології, неправильне чи неефективне розподілення природних ресурсів, та намагаються підвищити ефективність використання при мінімальних затратах.

Із розвитком технологій все більше процесів та процедур починають автоматизовувати, оскільки це значна економія часу та сил. Сфера енергоменеджменту не є виключенням – все частіше рутинні речі, які раніше робила людина, починають робити машини. Це дозволяє уникнути помилки при аналізі даних, заощадити час на побудові графіків та гістограм, що необхідні для аналізу, та зосередитися на вирішенні проблеми – як ефективніше використовувати ресурси.

Створення систем енергомоніторингу є основною складовою у запровадженні програми енергоменеджменту. Впровадження систем енергомоніторингу дає великий приріст у ефективності використання енергетичних ресурсів, економію ресурсів та бюджету на них, та оптимізацію експлуатації комунікаційних мереж.

Проаналізувавши інформацію із мережі Інтернет, стало зрозуміло що на даний момент існує дуже мало універсальних рішень чи платформ, що дозволяють швидко створити систему енергомоніторингу під свої потреби.

Більшість таких систем, що впроваджуються на підприємствах для контролю енергоресурсів, розробляються на комерційній основі та недоступні широкому загалу, тому було прийнято рішення розробити власну систему, що може бути використана у НТУУ «КПІ».

Було запропоновано створити систему, що дозволить зберігати та обробляти інформацію про використання енергоресурсів, та температурні режими, з метою оптимізації їх використання.

Система повинна надавати можливість внесення та редагування даних про енергоресурси, корпуси та аудиторії, температури в них, надавати можливість аналізувати зібрану інформацію, та формувати звітну інформацію.

Головний енергоменеджер повинен мати можливість додавання користувачів у систему та контроль їх доступу до функціоналу за рахунок гнучкої ієрархії ролей доступу.

Для розробки системи було визначено потреби служби енергоменеджменту, проаналізовано предметну область та існуючі рішення, і запропоновано розробку нового рішення.

Метою роботи визначено розробку системи для служби енергоменеджменту, що забезпечить виконання таких завдань: облік інформації про корпуси та аудиторії, показники тепло лічильників та статичні дані лічильників, можливість формування та завантаження звітної інформації про температурні режими у аудиторіях.

Записка складається із 5 розділів.

Перший розділ містить опис задачі розробки системи, другий описує предменту область та проблематику задачі. Третій розділ описує засоби розробки, що були використані у роботі над проектом. Четвертий розділ містить опис реалізації продукту та архітектуру програми, п’ятий розділ містить опис роботи користувача із системою.

1. ЗАДАЧА РОЗРОБКИ СИСТЕМИ ЕНЕРГОМЕНЕДЖМЕНТУ

Система енергоменеджменту автоматизує обробку та аналіз даних про використання енергоресурсів та температурні режими будівель. Основне її призначення – підвищити ефективність та раціональність використання ресурсів, надати можливість аналізувати температури в приміщеннях НТУУ «КПІ» , що дозволить швидко виправляти існуючі недоліки у енергоефективності будівель, та спостерігати за дотриманням норм температур.

1.1. Потенційні користувачі системи

Система може використовуватися службою енергоменеджменту НТУУ «КПІ» : головним енергоменеджером, та відповідальними особами у кожному корпусі. Головний енергоменеджер має можливість додавати користувачів із певними правами доступу до системи.

В залежності від рівня доступу, користувач може переглядати інформацію, формувати звіти, чи вносити/редагувати певні дані. Ролі користувачів поділені відповідно до модулів, та кожна наступна роль включає в себе функціонал ролі із нижчим рівнем доступу.

1.2. Загальний опис функціоналу системи

Відповідно до потреб служби енергоменеджменту, було поставлено задачу розробити систему, що буде містити необхідну інформацію про корпуси та аудиторії. Цю інформацію може вносити та редагувати особа з відповідними правами доступу. Також, система міститиме інформацію про встановлені у корпусах лічильники, та надає можливості редагування як

статичної інформації про лічильники (номер, серія, тип, тощо), так і їх показники (залежить від лічильника).

Відповідальна особа може також вносити інформацію про температури у аудиторіях, отримувати інформацію про температури. Система також може формувати звіти у форматі Excel.

Головний енергоменеджер має найбільше прав у системі, і може додавати нових користувачів, та змінювати їм права доступу.

1.3. Особливості розробки системи

Систему було розбито на підсистеми, згідно принципу клієнт-серверної архітектури. У якості індивідуального завдання було поставлено задачу розробити серверну частину додатку, та передбачити взаємодію через API.

Клієнт-серверна архітектура дозволяє розбити програму на логічні компоненти, які нічого один про одного не знають: сервер зберігає та обробляє дані, клієнт – відображає їх у зрозумілому для користувача вигляді.

Частини системи взаємодіють між собою за допомогою HTTP запитів, у якості параметрів запиту та відповіді використовується формат JSON. Використання цього формату обумовлено простотою серіалізації/десеріалізації даних. Фактично, використання цього формату стало стандартом у побудові програмного забезпечення із розділенням серверної частини та клієнтської.

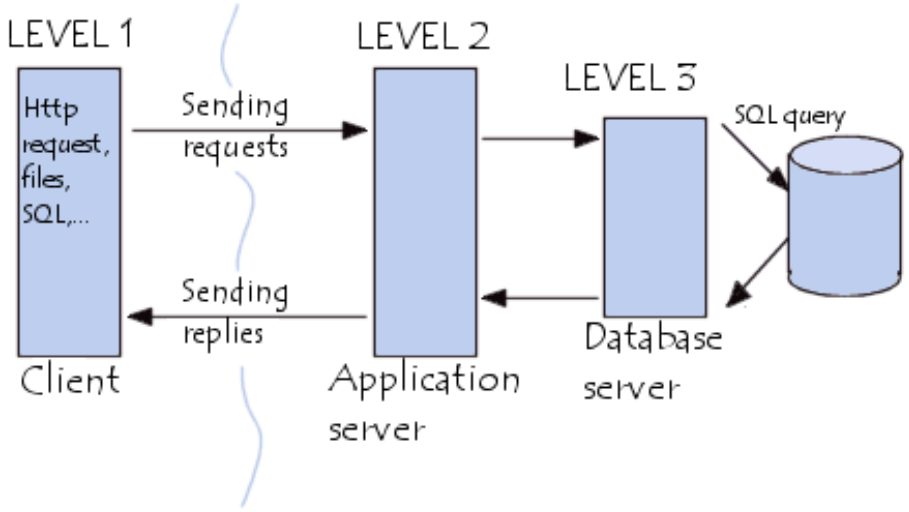

Рис.1.1 Схема архітектури «клієнт-сервер»

Архітектура «клієнт-сервер» набула широкого поширення у сучасній розробці завдяки розділенню обов'язків - серверна частина містить у собі бізнес-логіку, та зберігає дані, клієнтська частина – лише надсилає запити та відображає дані, при цьому клієнтів може бути кілька, і всі вони можуть бути написані за допомогою різних технологій.

Така схема взаємодії також дозволяє легко підтримувати та модифікувати код лише там, де це потрібно, єдине що лишається незмінним – структура запитів та відповідей. Це дає можливість писати більш «чистий код», оскільки дані потрібно віддавати у максимально простому та зрозумілому вигляді.

2. АНАЛІЗ ПРОБЛЕМИ ЕНЕРГОМЕНЕДЖМЕНТУ

У дипломній роботі було проведено аналіз проблеми енергомоніторингу, досліджено актуальність цієї проблеми та проаналізовано існуючі рішення у цій сфері.

2.1. Проблеми використання енергоресурсів

Із зростанням вартості енергетичних ресурсів та накопиченням екологічних проблем, що пов’язані з їх використанням, розвинуті країни почали задумуватися про зміни у цій сфері. Існують два очевидні рішення, що дозволяють знижувати вартість та шкідливість енерговикористання для навколишнього середовища.

Перше - перехід на альтернативні види енергії. Відмова від традиційних способів (вугілля, нафта), та нарощування використання відновлюваних видів енергії, таких як сонячні та вітрові електростанції не може відбуватися швидко. І не всюди можливо видобувати стільки альтернативної енергії, щоб забезпечити потреби населення.

Друге – оптимізація використання ресурсів. Якщо виявити де енергоресурси використовуються неефективно, та підвищити ефективність, можна значно скоротити їх витрати. І саме для цього було створено системи енергоменеджменту.

2.2. Поняття систем енергоменеджменту

Перш за все, необхідно визначити що таке енергоменеджмент та енергомоніторинг.

В загальному розумінні енергоменеджмент – це система керування енергоресурсами та їх використанням, а енергомоніторинг – реєстрація та

контроль енерговикористання. Тобто, фактично, енергоменеджмент включає в себе енергомоніторинг та будується на його основі.

Енергомоніторинг дає розуміння про стан технічного обладнання та його експлуатаційні можливості, зберігає та обробляє дані, що необхідні для моніторингу, та дозволяє оперативно приймати рішення що сприятимуть оптимізації використання ресурсів.

Енергоменеджмент неможливий без системи енергомоніторингу. На її основі енергоменеджер виявляє та аналізує існуючі проблеми у енергоспоживанні. Виходячи з отриманих даних розробляється план по скороченню витрат енергоресурсів, бюджету та підвищенню ефективності експлуатації комунікаційних систем.

В найпростішому варіанті системою енергомоніторингу може бути і звичайний блокнот, у якому ведеться запис показників лічильників, температур у приміщенні та інших величин, що характеризують енергоефективність приміщення, чи будівлі.

Проте, очевидно що такий спосіб може підійти лише для моніторингу невеликої площі, наприклад квартири чи будинку, та абсолютно не підходить для контролю енергоспоживання цілим підприємством чи, наприклад, університетом.

2.3. Енергоменеджмент в Україні та світі

Розвинуті країни давно почали запроваджувати в себе подібні системи для установ чи навіть цілих населених пунктів. Не в останню чергу це було пов’язано із зростанням цін на енергоносії, що змусило замислитися про нові підходи до використання енергії.

Найближчим до України сусідом, що успішно впровадив такі системи є Хорватія. На розробку системи було витрачено біля 600 тисяч доларів, проте результати її роботи були вражаючими – економія до 20 мільйонів євро на

рік. Цей приклад яскраво ілюструє наскільки вигідною є оптимізація використання енергоресурсів.

В Україні подібні системи поки що не набули великого поширення. В основному вони впроваджуються комерційними підприємствами, що хочуть зменшити свої витрати, або окремими ініціативними групами у рамках району/міста.

Як показують дослідження, виключно за рахунок використання системи енергомоніторингу (без масштабної реконструкції та впровадження енергоменеджменту) можна зменшити споживання енергії у містах до 10% - за рахунок різного попиту на енергетичні ресурси впродовж дня.

Тому проблема впровадження таких систем є дуже актуальною на сьогоднішній день.

2.4. Аналіз існуючих рішень

Із зростанням інтересу до енергоменеджменту зростає і кількість систем, що дозволяють автоматизувати цей процес. Проте, більшість цих систем є комерційними. Розглянемо дві системи, що знаходяться у відкритому доступі, та мають конкретний опис своїх можливостей

2.4.1 Автоматизована система енергомоніторингу

Автоматизована система енергомоніторингу (далі АСЕМ) – комплекс технічного та програмного забезпечення, що забезпечує дистанційний облік використання паливно-енергетичних ресурсів.

АСЕМ отримує дані безпосередньо від вузлів обліку електроенергії, тепла, холодної води, та збір інформації про температури у приміщеннях. Її основне призначення – вирішення питань контролю, раціоналізація використання ресурсів та підвищення ефективності споживання.

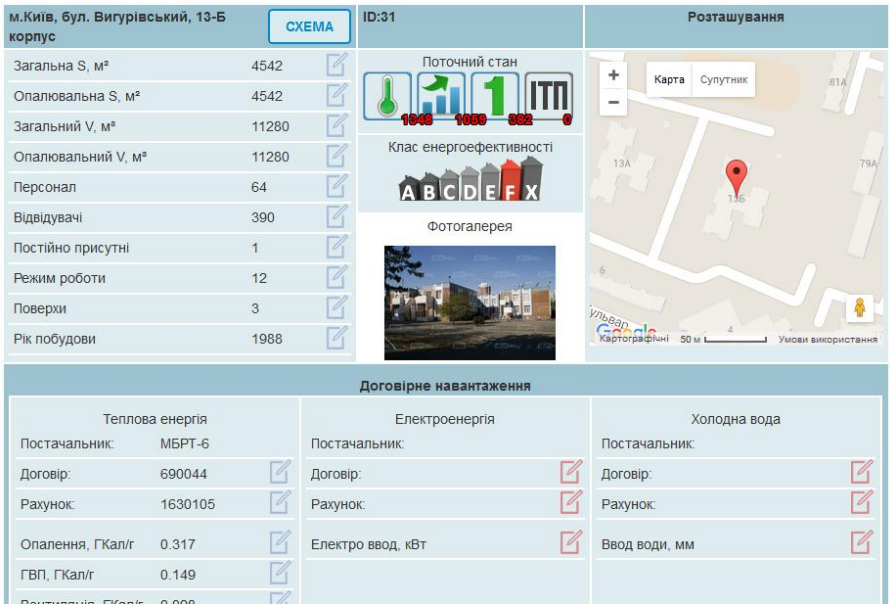

Рис.2.1. Інтерфейс АСЕМ

АСЕМ підтримує пропонує наступний функціонал:

1. Збір даних в автоматичному та ручному режимах
2. Можливість розширення системи
3. Можливість підключення додаткових модулів
4. Автоматизований аналіз даних
5. Візуалізація режимів роботи обладнання
6. Розмежування прав доступу користувачів.

2.4.2. Інформаційна система енергомоніторингу

Інформаційна система енергомоніторингу (далі ІСЕ) – система енергомоніторингу із великою кількістю потужних налаштувань. Вона є відкритою для інтеграції всіх систем, та взаємодіє із іншими системами та користувачами за допомогою API.

Розроблена на продуктах Symfony та Ember. Одна з головних переваг полягає у тому, що вся інформація зберігається на віддалених серверах, і

користувачі можуть використовувати її практично з будь-якого пристрою із виходом у мережу Інтернет.

Підтримує необмежену кількість доданих пристроїв обліку,фактично за допомогою цієї системи можна впровадити енергомоніторинг не лише для конкретної установи, а і для цілого міста. Також підтримує ручний, напівавтоматизований, та автоматизований засоби вводу даних у систему, поділ користувачів на ролі, та ін.

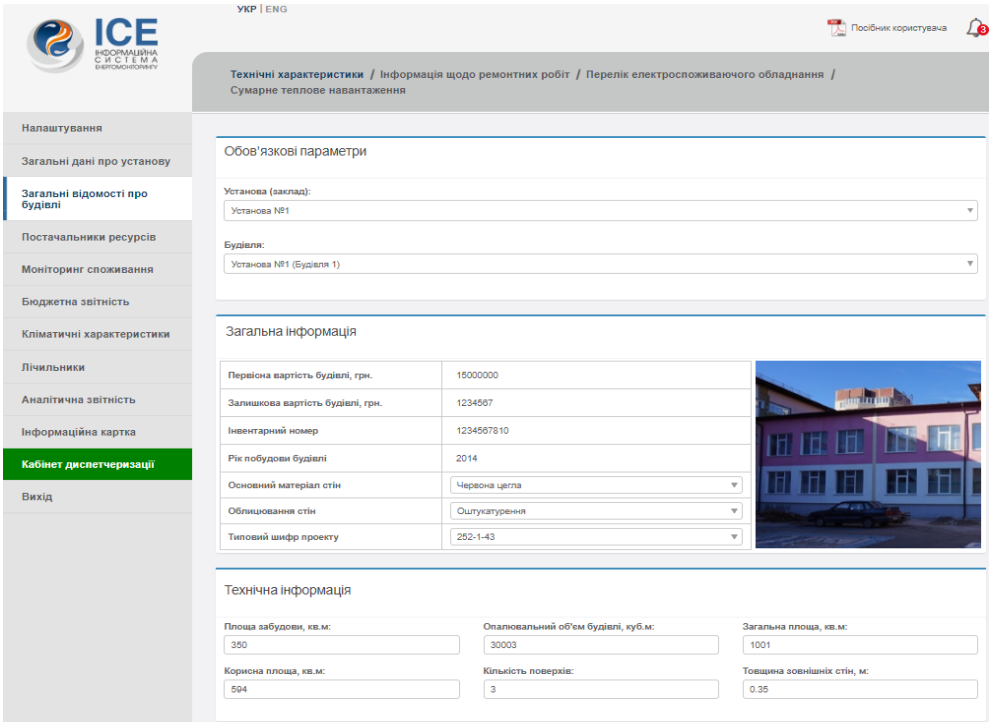

Рис.2.2. Інтерфейс вводу технічних характеристик будівлі.

На рис.2.2. зображено процес додавання будівлі у систему, її технічної інформації що згодом буде використана при обчисленнях та розрахунках. Кожна будівля підтримує додавання лічильників різних типів до неї. Заповнення показників лічильника може бути як автоматичним (напряму із лічильників), ручним (вводить відповідальна особа), так і напівавтоматичним (імпорт із файлів Excel).

Програмне забезпечення також здатне будувати звітність (аналітичну та бюджетну).

2.5. Висновки до розділу

У розділі було проведено аналіз поняття систем енергомоніторингу, та поняття енергоменеджменту, призначення систем та актуальність їх використання. Розглянуто існуючі системи, їх переваги та недоліки. На основі зроблених висновків було сформовано бачення того, як повинна виглядати розроблена система та передбачено можливість розширення програмного продукту у майбутньому.

3. ЗАСОБИ РОЗРОБКИ

Для розробки програмного забезпечення, а саме його серверної частини, було обрано наступні технології: мова програмування PHP, фреймворк Laravel, система керування базами даних MyS QL, та бібліотека PHPSpreadsheet для реалізації роботи із Excel форматом.

Також було обрано такі засоби розробки: середовище розробки PHPStorm 2019.1, система контролю версій Git, програма для тестування API – Postman.

3.1. Середовище розробки PHPStorm

PHPStorm – середовище розробки для PHP, розроблене компанією JetBrains. За основу було взято платформу IntelIJ IDEA.

Редактор надає можливість роботи із сучасними версіями мови PHP, підтримує також JavaScript, HTML та CSS. Можлива підтримка інших мов, діалектів та форматів за допомогою встановлення плагінів.

Середовище розробки надає можливість підключення до бази даних для забезпечення зручної розробки системи, із всіма необхідними функціями (створення таблиць та баз, редагування, тощо).

Величезною перевагою, у порівнянні з іншими редакторами є інтелектуальний аналіз коду, за допомогою якого середовище розробки може вказувати на повтори коду, допомагати скорочувати зайві операнди та оператори, зайве використання змінних, автоматично доповнювати код, та ін.

Ці можливості допомагають розробнику зосередитися на розробці кодової бази та бізнес – логіки. Відповідно до сучасних стандартів розробки, редактор підтримує інтеграцію із системами контролю версій, значно спрощуючи контроль за змінами в коді програмного продукту.

PHPStorm було розроблено у 2009 році, і з того часу середовища розробки зробили великий крок уперед у своїх функціональних можливостях. Дуже важливою особливістю цього середовища розробки є вбудована підтрика різних дебагерів для PHP.

Дебагер – програма для тестування та пошуку помилок у програмному коді продукту. Історично склалося так, що у PHP немає вбудованого де багера, тому для відладки коду потрібно використовувати сторонні рішення (напр. XDebug).

3.1. Система контролю версій Git

Система контролю версій – система що записує історію змін конкретних файлів, та дозволяє повертатися до конкретної версії цих файлів. Використання систем контролю версій обумовлено необхідністю мати історію розробки, та мати можливість у разі помилки повернутися до останньої правильно працюючої версії програми.

Також, у командній розробці системи контролю версій дозволяють команді швидко кооперуватися, та бачити зміни інших розробників, виконуючи всього лише кілька команд.

Існує багато систем контролю версій, і на сьогоднішній день найпоширенішою серед них є система Git, що стала де-факто стандартом у сучасній розробці.

Система Git – розподілена система контролю версій. Ця система повністю клонує репозиторій, а не лише зберігає історію змін. Таким чином вона практично унеможливлює втрату всіх даних, оскільки на кожному пристрої, де велася розробка, зберігається повна історія роботи над програмним забезпеченням.

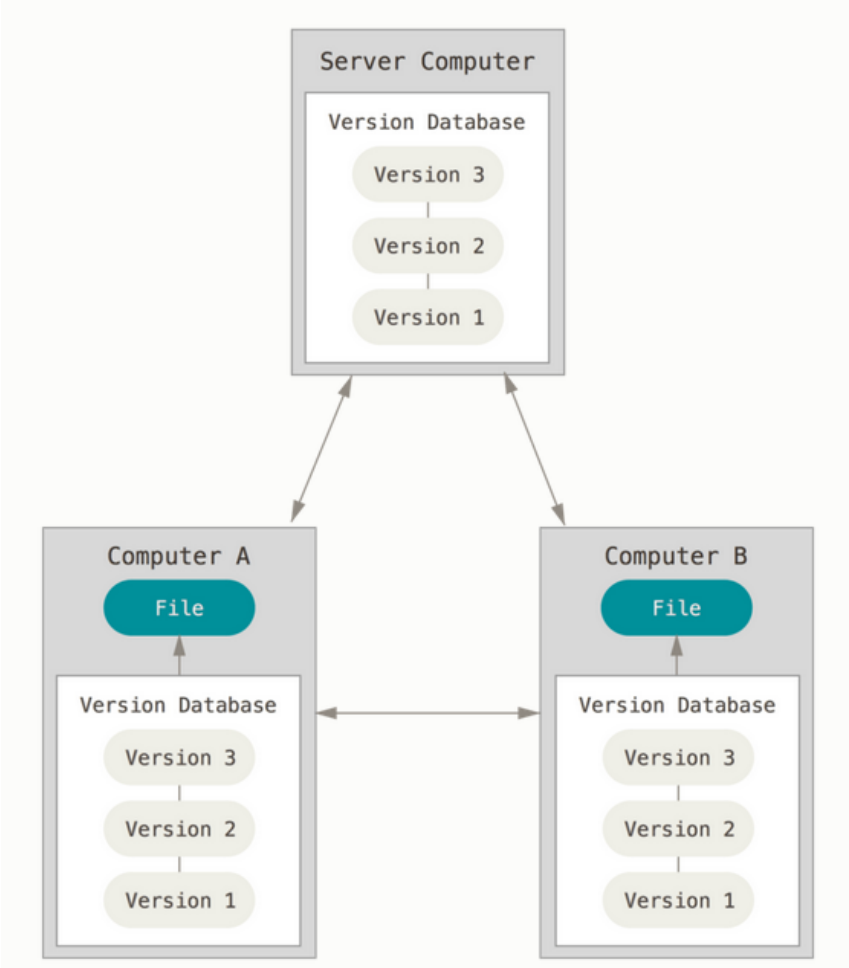

Рис. 3.1. Розподілений контроль версій.

Однією з переваг розподілених систем контролю версій є можливість взаємодії одночасно з багатьма віддаленими репозиторіями, таким чином програміст може працювати одночасно з кількома групами людей, застосовуючи різні підходи у розробці.

Основна відмінність Git від інших систем контролю версій – підхід до роботи з даними. Більшість систем зберігають інформацію у вигляді списку змін у файлах, тоді як Git «запам'ятовує» стан проекту під час кожного коміту. Це більше схоже на зберігання мініатюрних копій файлів проекту.

Робота із Git можлива у консольному режимі (рис.3.2), що дозволяє глибше зрозуміти принцип роботи системи, та у режимі окремого додатку з інтерфейсом, або ж, якщо середовище розробки підтримує таку можливість – як вбудований плагін у нього.

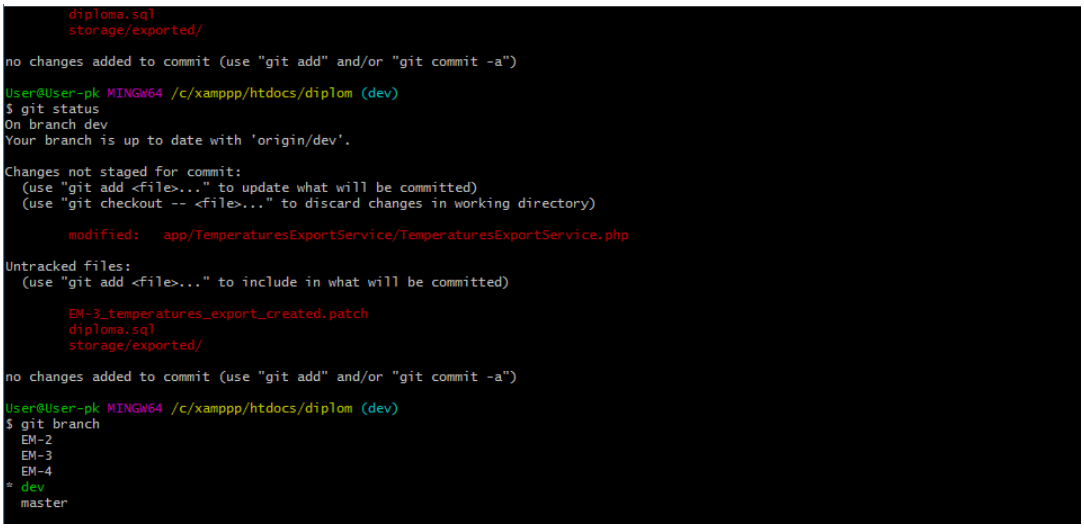

Рис.3.2. Консольний режим Git

Існують кілька різних варіантів розробки продукту із використанням Git в залежності від кількості людей у команді, типу продукту, та ін.. Для розробки дипломного проекту було обрано модель розробки git flow (рис.3.3).

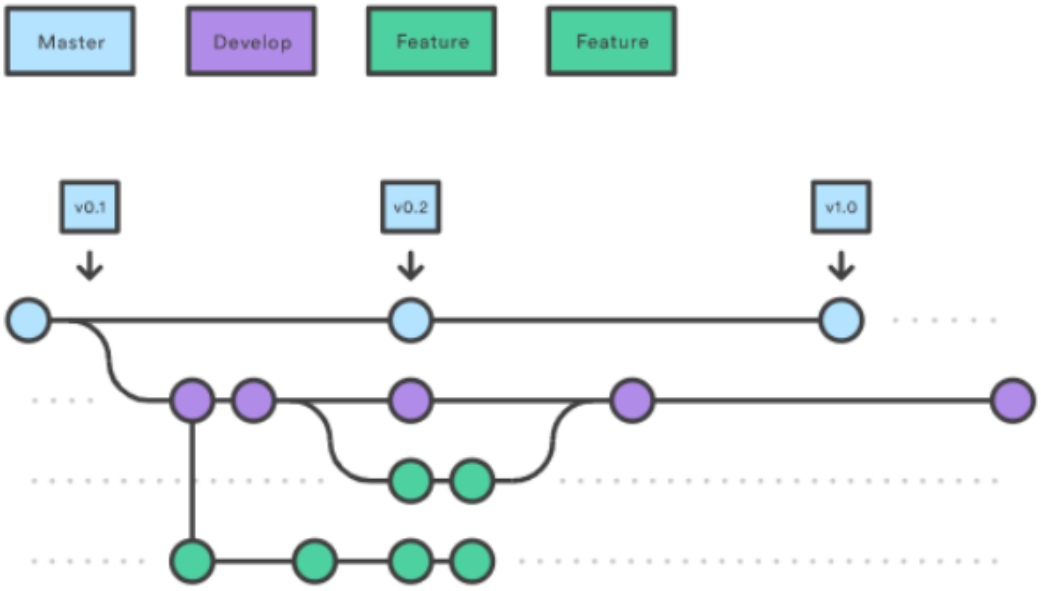

Рис.3.3. Модель розробки git flow

Відповідно до цієї моделі, проект має основну гілку – master, у якій знаходиться актуальна робоча версія програми. Гілка dev містить у собі версію програми, що на даний момент знаходиться у розробці, проте недостатньо протестована для того, щоб надавати її у широке користування.

Побудова програми повинна бути розбита на логічні під завдання, для кожного з яких створюється окрема гілка (наприклад під задача розробки експорту даних у Excel). Після виконання задачі вона вливається у гілку dev та тестується разом із іншими задачами.

Такий підхід для розробки дозволяє чітко розкласти розробку програмного продукту на логічні завдання, і працювати над ним як у команді, так і самому.

3.2. Програма для тестування API Postman

Програма Postman дозволяє швидко протестувати роботу будь-якого API, за допомогою гнучких та потужних налаштувань для запитів. Величезною перевагою, яку надає ця програма є можливість протестувати запити будь-яких типів, із будь-якими параметрами так, ніби цей запит відправив клієнтський додаток.

Програма підтримує всі доступні на сьогоднішній день типи запитів (GET, POST, PUT, тощо), дозволяє гнучко налаштовувати параметри запиту та заголовки.

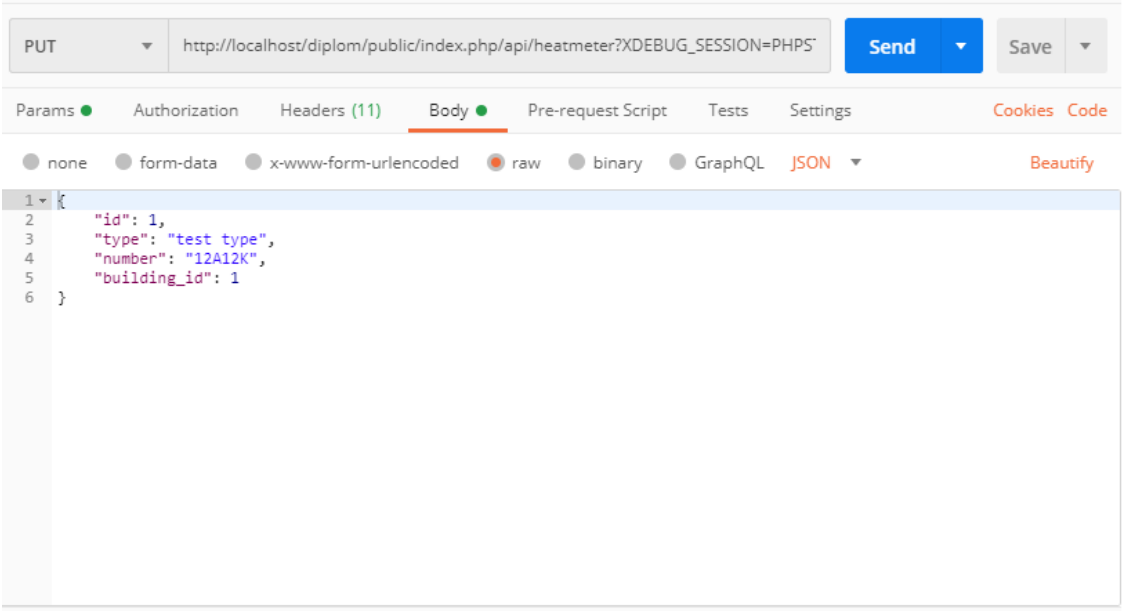

Рис.3.4. Приклад надсилання запиту в програмі Postman

Postman також підтримує збереження запитів у колекції, автоматичні тести, та документацію API відповідно до надісланих запитів.

Важливою особливістю програми є підтримка всіх сучасних стандартів авторизації та аутентифікації користувачів (напр. OAuth2), що дозволяє тестувати програмні продукти із різними сценаріями роботи для авторизованих користувачів.

3.3. Мова програмування PHP

Для серверної частини проекту було обрано мову програмування PHP. Це зумовлено тим, що PHP - найбільш популярна мова для розробки серверної частини проектів, фактично – це мова яка була створена для веб-розробки. Робота з HTTP запитами у ній підтримується без встановлення додаткових бібліотек.

Також важливою особливістю цієї мови є те, що програми, написані нею, не виконуються постійно, а лише по зверненню до сервера. Тобто, користувач надіслав запит, сервер віддав відповідь, робота програми закінчилася. Це дозволяє оптимізувати час роботи сервера, сервер менше завантажується, та швидше працює.

PHP є однією з найпоширеніших мов програмування у світі, та набільш популярною мовою для програмування серверної частини ПЗ, що працює у мережі Інтернет. Розроблена в 1995 році, вона пройшла довгий шлях еволюції від невеликого інтерпретатора шаблонів, до повноцінної С-подібної мови із підтримкою ООП, великою екосистемою, стандартами розробки та всіма сучасними практиками програмування.

У перших своїх версіях PHP був досить простою мовою програмування, проте швидко набув популярності саме завдяки цій простоті, та у більшій мірі задовольняв потреби програмістів.

Із зростанням використання PHP у якості основної мови програмування, росли і вимоги до неї. Оскільки над першими версіями працювали всього кілька розробників, у роботі мови зустрічалися різного роду помилки, вона була не дуже добре оптимізована, а також не підтримувала ООП, що дуже ускладнювало процес розробки великого за своїми розмірами програмного забезпечення.

У PHP 5 нарешті було додано підтримку ООП, і з того часу екосистема PHP значно еволюціонувала. Програмне забезпеченя стало легше писати та підтримувати, а із виходом PHP 7, у якій додали підтримку типізації, покращили роботу генераторів, оптимізували роботу самої мови, PHP практично нічим не поступається найбільш поширеним строго типізованим мовам (Java, C# тощо) як за можливістю будувати програмне забезпечення з усіма сучасними стандартами, так і за швидкістю роботи коду.

Для стандартизації розробки на PHP існують рекомендації по написанню коду: PSR. Ці стандарти описують те, як повинен виглядати код на PHP, щоб забезпечити його читабельність, та можливість підтримки коду іншими розробниками. Відповідно до цих стандартів побудовані всі сучасні фреймворки та бібліотеки.

3.5. Фреймворк Laravel

У сучасній розробці прийнято використовувати вже робочі інструменти, для покращення якості ПЗ. Щоб зосередитися на розробці бізнес-логіки, та абстрагуватися від роботи таких речей, як наприклад, обробка запитів, роутинг та ін., використовуються фреймворки.

З цією метою було обрано фреймворк Laravel як один із найпотужніших та найбільш гнучких фреймворків для мови програмування PHP. Його структура проста та зрозуміла, заснована на шаблоні MVC (Model-view-controller). Фреймворк підтримує вбудовану авторизацію та

аутентифікацію користувачів, утиліти для роботи із БД, повноцінну ORM Eloquent, та багато іншого.

Найбільш поширеними серед фреймворків на PHP є два фреймворки: Laravel та Symfony. Популярності вони набули завдяки компонентному підходу до розробки, потужним можливостям для розширення та додавання функціоналу.

Багато компонентів у цих фреймворках є спільними, проте є і ключові відмінності: Laravel більш «легкий» фреймворк, на відміну від Symfony, який призначений більше для командної розробки великих Enterprise проектів.

Проте,модульна структура дозволяє легко замінити один компонент фреймворку на будь-який інший, наприклад для роботи із БД можна використовувати ORM Doctrine (що використовується в Symfony).

Також, можна додавати або перевизначати компоненти фреймворку із власним функціоналом, що забезпечує велику гнучкість у налаштуваннях.

Завдяки всім цим якостям, Laravel є дуже поширеним фреймворком на базі якого написані багато проектів, і навіть цілі CMS.

3.4. Система керування базами даних MySQL

Для зберігання даних у багатьох додатках використовуються системи керування базою даних.

База даних (БД) – упорядкований набір логічно взамємопов’язаних даних. Система керування базою даних (СКБД) - комплекс програмних і мовних засобів, необхідних для створення баз даних, підтримання їх в актуальному стані та організації пошуку в них інформації.

Головним завданням БД є гарантоване збереження великих об’ємів інформації та надання доступу до неї користувачеві/іншій програмі.

Для взаємодії із БД використовується мова SQL.

У курсовій роботі використовується СКБД MySQL. Це вільна система керування реляційними базами даних, яка була створена як альтернатива комерційним системам. На сьогоднішній день це одна з найпоширеніших систем керування базами даних, оскільки вона має велику швидкість, підтримку з боку більшості мов програмування, та є простою у встановленні та використанні.

Переваги MySQL:

1. Просте використання та встановлення
2. Висока швидкість
3. Потужна система безпеки
4. Велика (до 50 млн) кількість рядків у таблицях
5. Відкритий код.
6. Працює із будь-якою ОС

4.ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ

Відповідно до індивідуального завдання, було розроблено серверну частину системи енергоменеджменту, що надаватиме API для зберігання та обробки інформації.

Для розробки серверної частини було обрано мову програмування PHP та фреймворк Laravel. Обрані технології дозволили побудувати чітку та зрозумілу архітектуру системи, із дотриманням всіх сучасних стандартів та кращих практик розробки.

У основу фреймворку Laravel було покладено патерн MVC, що дозволяє розділити логіку обробки даних, логіку роботи з базою даних, та логіку представлення. Проте, оскільки за мету було поставлено розробку API, логіка представлення як така у системі практично відсутня.

4.1. Опис роботи із базою даних

Як було зазначено вище, у програмному продукті було застосовано розділення логіки обробки даних та логіки взаємодії з базою. У найпростішому вигляді це можна реалізувати наступним чином: при отриманні запиту ззовні, викликається контролер, який в своїх методах викликає модель, та працює з нею. Проте, цей підхід є не зовсім правильним, і, наприклад, у разі зміни логіки зберігання даних, доведеться робити більше змін. Окрім того, такий підхід суперечить принципу Low coupling, оскільки контролери «жорстко» пов’язані із класами моделі. Тому, при розробці системи було використано патерн «Репозиторій».

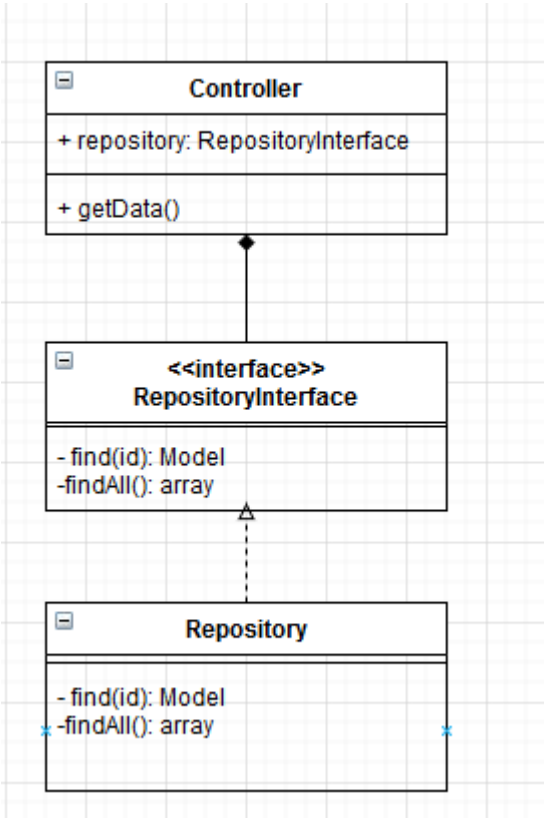

Рис.4.1. Патерн «Репозиторій»

Цей патерн дозволяє приховати логіку роботи із базою даних, та класами моделі всередині спеціального класу – репозиторія. Окрім того, що важливо – контролер не прив’язаний до репозиторію «жорстко» - він містить у собі інтерфейс.

Цей інтерфейс може бути реалізований багатьма репозиторіями, і в залежності від ситуації контролер буде отримувати той чи інший об’єкт, що реалізує цей інтерфейс.

Всередині репозиторій використовує класи моделі – об’єктно орієнтоване представлення таблиць у БД. Для розробки класів моделі використовувалася ORM Eloquent.

4.2. Опис бази даних

Для роботи системи, зберігання та зчитування даних було розроблено модель БД, що зображена на рис.4.2.

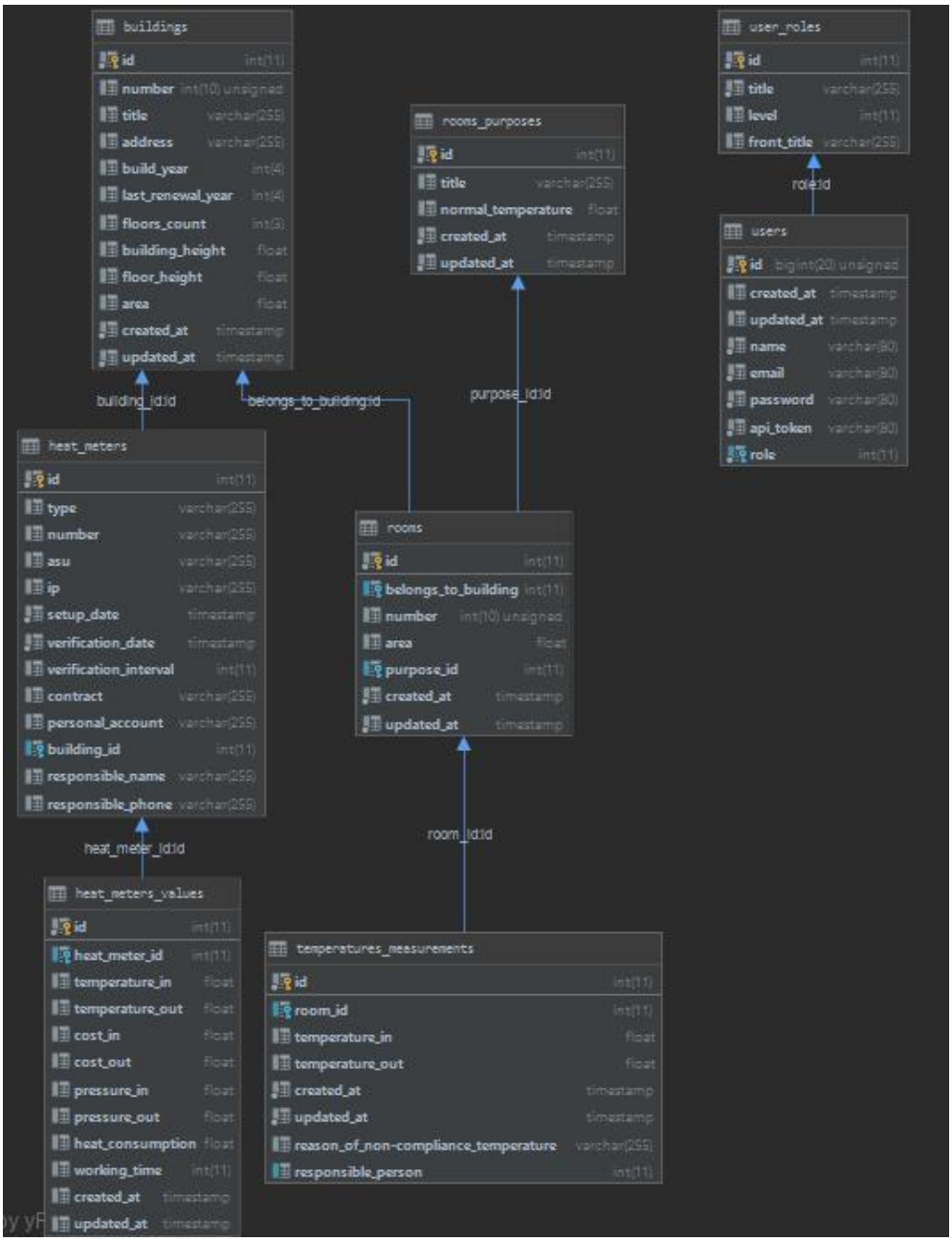

Рис.4.2. Модель бази даних системи

Відповідно до вимог системи, було розроблено модель БД, що зберігатиме усі необхідні дані – інформацію про корпуси, аудиторії, користувачів системи та їх ролі. Також, були створені таблиці, що містять у

собі інформацію про заміри температур у аудиторіях, та таблиці із даними теплолічильників – статичною інформацією про сам теплолічильник, та таблиця із показниками.

В окрему таблицю було винесено інформацію про типи аудиторій та нормативні температури у них.

4.2. Опис функціональних можливостей системи

На рис.4.2. зображено акторів (користувачів системи) та доступні їм дії

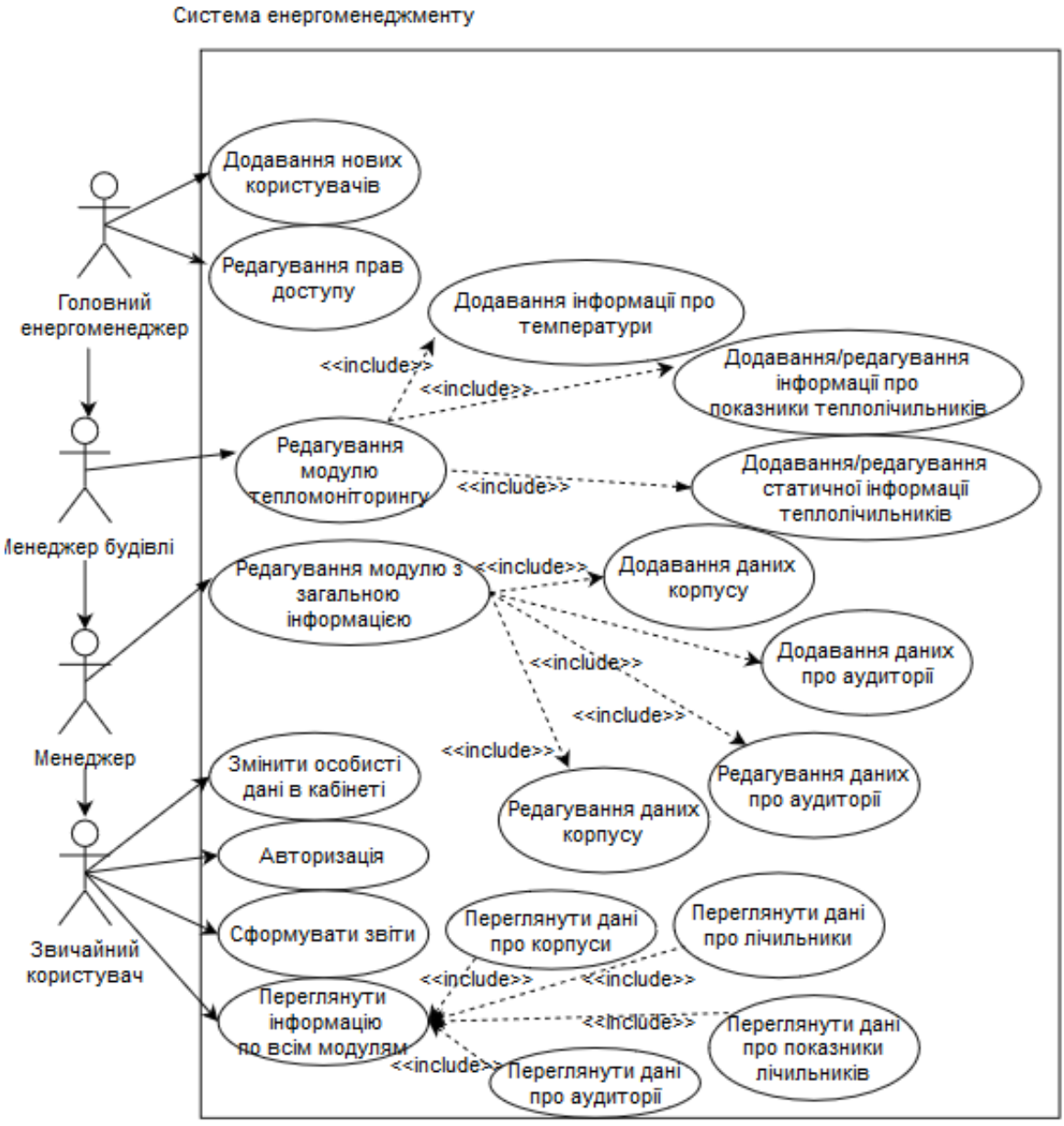

Рис.4.2. Діаграма прецедентів системи

Розглянемо один із можливих сценаріїв взаємодії користувача із системою на прикладі запиту про корпуси. Процес роботи системи у спрощеному вигляді можна проілюструвати за допомогою діаграми послідовностей (з англ. Sequences diagram), що зображена на рис.4.3

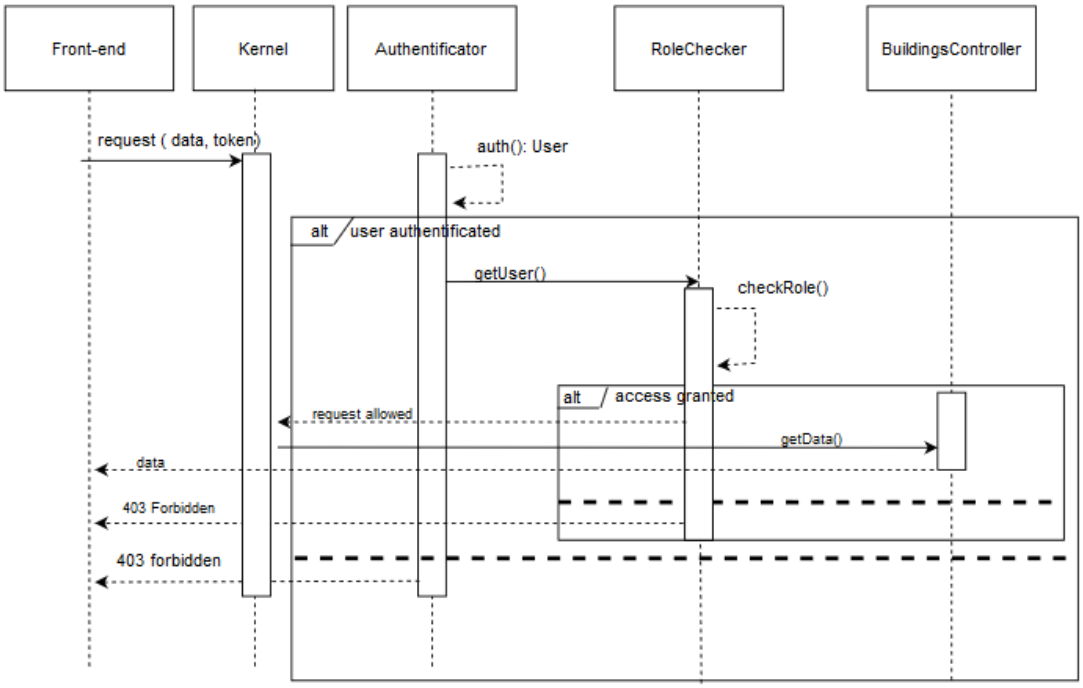

Рис.4.3. Діаграма послідовностей для запиту інформації про будівлі

На діаграмі зображено ключові елементи роботи системи: коли із клієнтської частини додатку (Front-end) надходить запит, його приймає на обробку класс Kernel. Це системний клас фреймворку Laravel, що відповідає за обробку запитів. Він по черзі віддає запит на перевірку ланцюжку допоміжних класів (middlewares), і у разі якщо помилок не виявлено – запит передається у потрібний контролер.

У розробленій системі основними middleware-класами є Authentificator та RoleChecker. Вони відповідають за аутентифікацію та перевірку прав доступу користувача відповідно. На кожному з етапів у разі якщо виявлено помилку (неправильний токен авторизації/недостатньо прав доступу) вони повертають відповідь “403 Forbidden”.

403 помилка – це стандартний код помилки, у разі коли користувач/стороннє API не має прав доступу до дії/інформації.

Аналогічно до обробки цього запиту, система обробляє абсолютно всі вхідні запити (за винятком запиту авторизації/виходу із системи). Подібна структура дозволяє один раз написати логіку перевірок прав доступу, і надалі зосередитися на розробці контролерів що безпосередньо відповідають за обробку даних.

4.3. Ролі та права користувачів у системі

Відповідно до індивідуального завдання, у системі було реалізовано ієрархію ролей. Кожен користувач системи відповідно до свого рівня доступу має певні права. Ролі користувачам назначає головний енергоменеджер.

Коли програма отримує запит, перед тим як виконувати обробку даних, відбувається перевірка, чи користувач що зробив цей запит, має на це право. Модуль перевірки ролей доступу користувача розроблено гнучким, із можливістю подальшого розширення ієрархії ролей.

В базі даних ролі список ролей зберігається у спеціальній таблиці (рис.4.3)

Рис.4.4. Таблиця з ролями користувачів

В ній зберігається назва ролі, та число – рівень доступу ролі. Відповідно, кожен наступний рівень може робити те, що доступно рівням нижче.

У кодовій базі перевіркою ролі користувача займається спеціальний сервіс. У разі, якщо рівень доступу користувача більше за необхідний для

виконання дії, дія виконується, якщо менше – відповідно повертається повідомлення про те, що користувачеві недоступний цей функціонал.

У разі ж коли рівень доступу до функціоналу рівний рівню доступу користувача, відбувається перевірка чи співпадають назви ролі. Це дозволяє розширювати ієрархію ролей, створюючи користувачів з рівним рівнем доступу, проте з доступом до деяких функцій, що доступні лише їм.

4.4. Структура проекту

Для розробки проекту та подальшої його підтримки необхідно правильно структуризувати систему, та розмістити файли проекту так, щоб їх можна було швидко знайти, тобто вони повинні бути логічно поділені на так звані пакети.

В PHP немає підтримки пакетів, їх роль виконують простори імен (namespaces). Кожен клас повинен належати до простору імен, що за своєю структурою відповідає структурі файлової системи проекту. Оскільки у дипломній роботі було використано фреймворк Laravel, у ньому вже визначена базова структура файлової системи, що зображена на рис. 4.5.

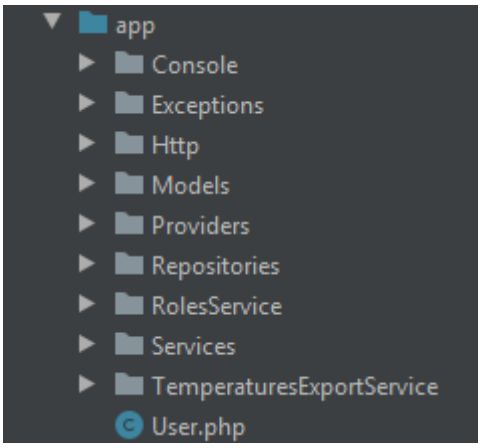

Рис.4.5. Коренева структура проекту

Коренева структура поділена на підпапки, у яких зберігаються контролери, сервіси та інші допоміжні класи.

У папці Http містяться контролери та допоміжні класи аутентифікації та перевірки ролей. (рис.4.6).

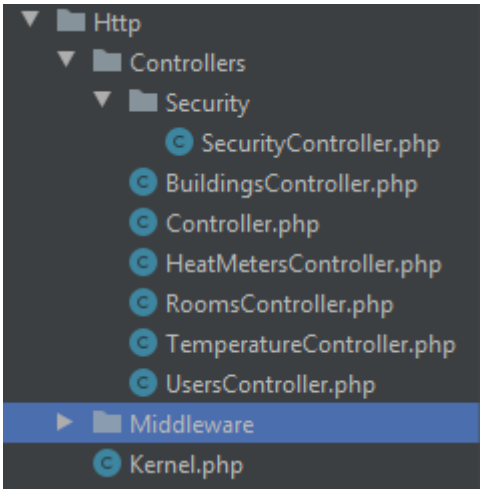

Рис.4.6. Контролери системи

Контролери при отриманні запиту, за потреби, виконують валідацію вхідних даних (наприклад, перевіряють чи не існує у системі заданого паролю). У разі простих операцій (робота із базою) контролери викликають репозиторії, що містяться у відповідній папці (рис.4.7). Принцип роботи патерна «Репозиторій» було описано у розділі 4.1.

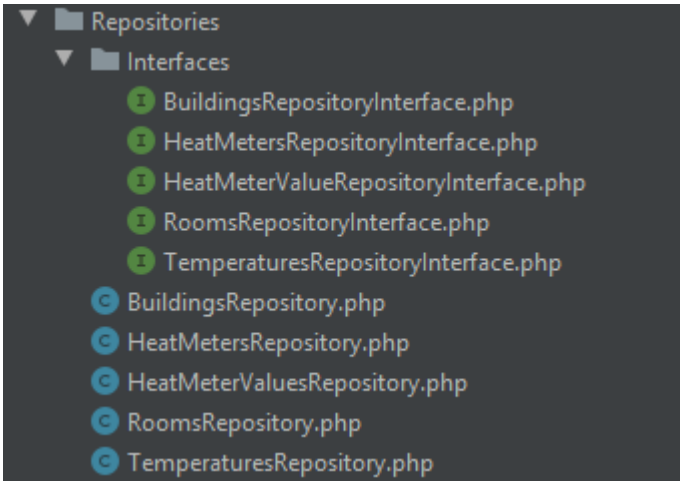

Рис.4.6. Класи-репозиторії проекту

Якщо ж необхідно провести більш складні операції (фільтрування даних, експортування), контролери не працюють напряму із репозиторіями, а викликають допоміжні сервіси, що приховують у собі бізнес-логіку

4.5. Модулі проекту

4.5.1 Модуль тепломоніторингу

В системі було реалізовано модуль тепло моніторингу із наступним функціоналом:

- Додавання/видалення/редагування інформації із тепло лічильників
- Перегляд даних із тепло лічильників
- Додавання/видалення/редагування інформації про температури в аудиторіях
- Перегляд інформації про температури у аудиторія
- Можливість завантаження звітної інформації

Звітна інформація завантажується у форматі Excel, і містить гнучкий набір фільтрів, за якими можна формувати звіти. Для роботи із форматом Excel використовувалася бібліотека PHPSpreadsheet, що має потужний набір інструментів для роботи із популярними форматами.

Переглядати інформацію та завантажувати звіти може будь-який авторизований користувач, а додавати/редагувати/видаляти – лише менеджер будівлі.

4.5.2 Модуль загальної інформації

Відповідно до завдання, в системі було розроблено модуль, що містить загальну інформацію про корпуси та аудиторії.

Цей модуль є необхідним для повноцінної роботи системи, оскільки інформація про температури, лічильники та їх показники прив’язуються до даних із цього модулю – аудиторій та будівель.

Модуль містить наступну функціонал:

- Перегляд інформації про корпуси (доступно всім користувачам)
- Перегляд інформації про аудиторії (доступно всім користувачам)
- Перегляд нормативної інформації про температури в залежності від типу аудиторії (доступно всім користувачам)

- Додавання/редагування/видалення інформації про корпуси (доступно менеджеру і ролям вище)
- Додавання/редагування/видалення інформації про аудиторії (доступно користувачам із роллю «Менеджер» і вище)

4.5.3 Особистий кабінет

Оскільки система доступна лише авторизованим користувачам, у них повинна бути можливість керувати своїми персональними даними (ім’я, адреса електронної скриньки, пароль тощо). Відповідно до цього, у особистому кабінеті користувачі мають можливість все це переглянути чи змінити.

У особистому кабінеті головного енергоменеджера є можливість переглядати загальну інформацію про користувачів, та, за потреби, змінювати їм ролі доступу. Також, головний енергоменеджер має можливість додати нових користувачів.

4. ОПИС РОБОТИ КОРИСТУВАЧА ІЗ СИСТЕМОЮ

5.1. Системні вимоги та інсталяція

Оскільки програмне забезпечення розроблялося як веб-додаток, користувачеві не потрібно встановлювати його. Для доступу до системи необхідно мати ПК або смартфон із доступом до мережі Інтернет та встановленим сучасним браузером із підтримкою мови Javascript (Google Chrome, Mozilla Firefox, тощо).

5.2. Сценарій роботи користувача із системою

При відкритті системи у браузері, користувач побачить форму входу у систему (рис.5.1.)

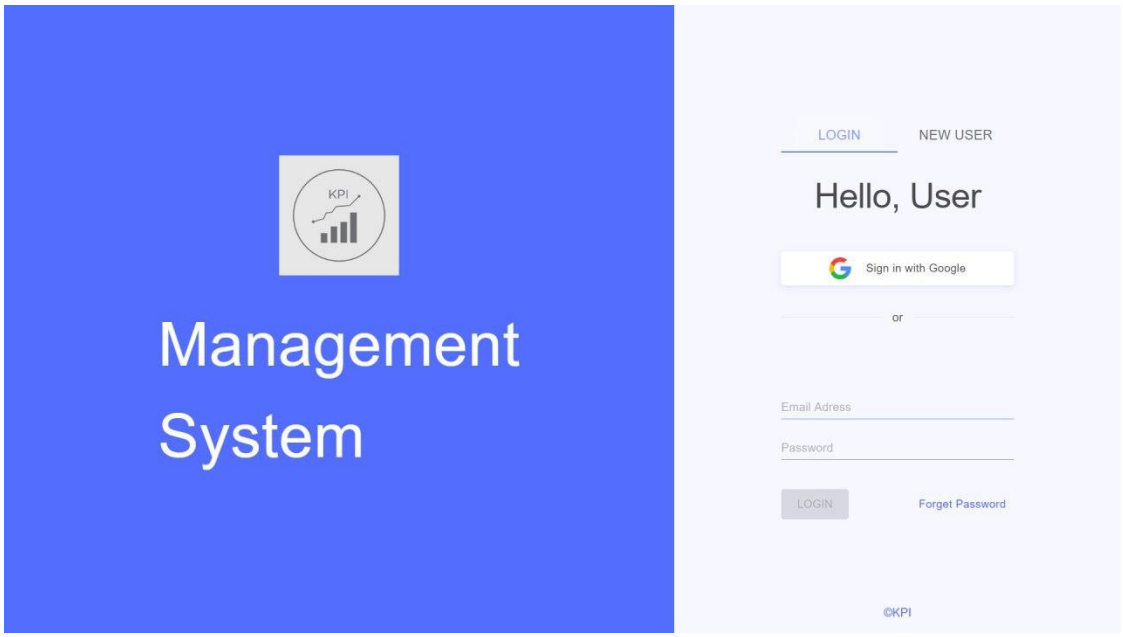

Рис.5.1. Сторінка входу у систему

Після того, як користувач увійде в систему, він потрапляє на головну сторінку, із якої може перейти у особистий кабінет, чи перейти до будь-якого із доступних йому модулів.

На лівій панелі відображаються модулі системи (рис.5.2)

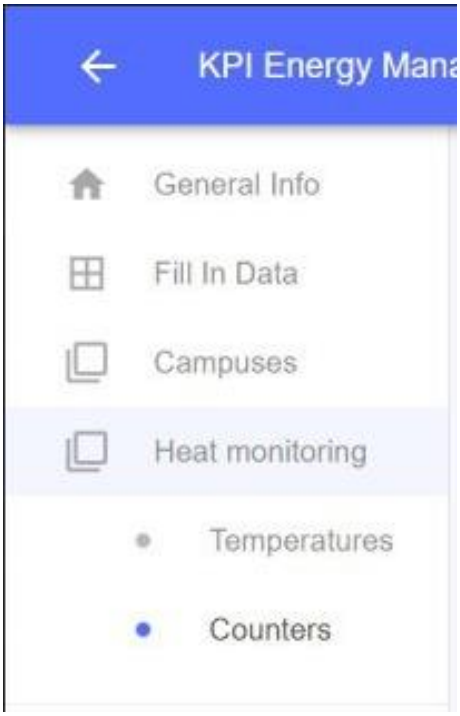

Рис.5.2. Меню системи

Будь-який користувач має можливість перейти та переглянути інформацію із цих модулів. Обравши пункт «температури» у модулі тепло моніторингу, користувач отримує можливість експортувати дані за заданими фільтрами (дати, проміжок температур, тощо) у формат Excel. Також, для користувачів із рівнем доступу «Менеджер будівлі» будуть доступні кнопки для внесення даних по температурам та редагування даних (рис.5.3)

Рис.5.3. Розділ «Температури»

При натисканні на кноку «Експортувати» система формує звіт у форматі Excel та завантажує його користувачеві. У звіті доступні наступні поля: Назва корпусу, номер аудиторії, температура у приміщенні, температура ззовні, нормативна температура для цього типу аудиторій, та відповідальна особа, що проводила вимірювання (рис.5.4).

Дата проведення в	Корпус	№ кімнати	Час проведення в	Фактична т	Нормативна темпе	Зовнішня температура	Відповідальна особа
		110	12:30:00	16		13	Бевза Д.В.
06.05.2020		111	19:22:21	14		12	
		112	18:11:39	15		11	
07.05.2020	Корпус 1	113	18:11:39	13	18	11	
		110	12:30:00	14		11	Бевза Д.В.
08.05.2020		111	19:22:21	16		16	
		112	18:11:39	15		15	
09.05.2020	Корпус 1	113	18:11:39	15	18	12	
		110	12:30:00	17		11	Бевза Д.В.
10.05.2020		111	19:22:21	18		17	
		112	18:11:39	19		18	
11.05.2020	Корпус 1	113	18:11:39	19	18	18	

Рис.5.4. Зразок сформованого звіту у форматі Excel

ВИСНОВКИ

Під час роботи над дипломним проектом було набуто навичок розробки серверної частини веб-додатків, набуто досвіду роботи у команді, та розробки систем згідно технічного завдання.

Розроблено систему, що відповідає поставленим вимогам.

Також покращено знання у області веб-програмування, а саме мови PHP. Вивчено можливості фреймворку Laravel, що є дуже хорошим досвідом для розробника, оскільки на сьогоднішній день використання фреймворків є стандартною практикою у сучасній розробці.

Покращенно навички роботи із системою контролю версій Git та GitHub, що є найпопулярнішою системою та використовується у багатьох IT-компаніях.

Вивчено можливості роботи із форматом Excel за допомогою бібліотеки PHPSpreadsheet, що є потужним засобом для формування не лише таблиць у форматі Excel, а й з іншими форматами.

Окрім цього, поглиблено знання та навички роботи із базами даних, а саме MySQL.

Отже, під час проходження практики було засвоєно та набуто достатню кількість знань, що необхідні сучасному веб-розробнику.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.PhpSpreadsheet documentation[Electronic resource]. — Access mode: https://phpspreadsheet.readthedocs.io/en/latest/

2.Ю.В.Хацкевич, Г.Г.Півняк. Системи енергоменеджменту та їх практичне забезпечення [Електронний ресурс] / Ю.В.Хацкевич, Г.Г.Півняк. — 2013. —215с. — Режим доступу: https://core.ac.uk/download/pdf/48403299.pdf.

3.Laravel documentation [Electronic resource]. — Access mode: https://laravel.com/docs/7.x/installation.

4.Robert C. Martin Clean code / Robert C. Martin – 2008 – 413с — O'Relly

5.PHP The right way [Electronic resource]. — Access mode: https://phptherightway.com/.

6.W3C Standerds HTML & CSS [Electronic resource]. — Access mode: https://www.w3.org/standards/webdesign/htmlcss .

Додаток А

Система збору даних про енергоресурси та температурні режими для служби енергоменеджменту

Специфікація

УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ6122_20Б

Аркушів 1

Київ — 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ6122_20Б	Пояснювальна записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ6122_20Б 12-1	Index.php, Controller.php	Основні компоненти, що запускають роботу інших
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ6122_20Б 13-1	Додаток В.docx	Опис програмного модуля
Комплекс		
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ6122_20Б	Серверна частина системи (backend)	Розроблялося в дипломній роботі
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ6122_20Б	Клієнтська частина системи (frontend)	

Додаток Б

Система збору даних про енергоресурси та температурні режими для служби енергоменеджменту

Текст програми

УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ6122_20Б

Аркушів 5

Київ — 2019

```
//модуль експорту температур в Excel або масив даних
class TemperaturesExportService implements TemperaturesExportServiceInterface
{
   private $conditions = [];

   /** @var TemperaturesRepositoryInterface */
   private $temperaturesRepository;

   private $inConditions = [];

   private $excelHeaders = [
      'date' => 'Дата проведення вимірювань',
      'title' => 'Корпус',
      'number' => '№ кімнати',
      'time'   => 'Час проведення вимірювань' ,
      'temperature_in' => 'Фактична температура',
      'normal_temperature' => 'Нормативна температура',
      'temperature_out' => 'Зовнішня  температура',
   ];

   private $columnsAllowedToMerge = [
      'date', 'title', 'normal_temperature'
   ];

   public function __construct(TemperaturesRepositoryInterface $repository)
   {
      $this->temperaturesRepository = $repository;
   }

   /**
    * @param array $requestedData
    * @throws WrongConfigureException
    */
   public function configure(array $requestedData)
   {
```

```
            if(isset($requestedData['date'])) {
        if(!is_array($requestedData['date'])) {
            if($this->validateDate($requestedData['date'])) {
                //date should be as yyyy-mm-dd hh:mm:ss
                $this->conditions[] = ['temperatures_measurements.updated_at', '>='
,$requestedData['date'].' 00:00:00'];
                $this->conditions[] = ['temperatures_measurements.updated_at', '<='
,$requestedData['date'].' 23:59:59'];
            }
        } else {
            if(!isset($requestedData['date']['from']) || !isset($requestedData['date']['to'])) {
                throw new WrongConfigureException('Wrong request configuration');
            }
            $this->conditions[] = ['temperatures_measurements.updated_at', '>=',
$requestedData['date']['from']];
            $this->conditions[] = ['temperatures_measurements.updated_at', '<=',
$requestedData['date']['to']];
        }
    }

    $this->inConditions = $requestedData['rooms'] ?? [];

    if(isset($requestedData['temperature'])) {
        if(isset($requestedData['temperature']['min'])) {
            $this->conditions[] = ['temperatures_measurements.temperature_in', '>=',
$requestedData['temperature']['min']];
        } else if(isset($requestedData['temperature']['max'])) {
            $this->conditions[] = ['temperatures_measurements.temperature_in', '<=',
$requestedData['temperature']['max']];
        } else if(isset($requestedData['temperature']['from']) &&
isset($requestedData['temperature']['to'])) {
            $this->conditions[] = ['temperatures_measurements.temperature_in', '>=',
$requestedData['temperature']['from']];
            $this->conditions[] = ['temperatures_measurements.temperature_in', '<=',
$requestedData['temperature']['to']];
```

```
            } else throw new WrongConfigureException('Wrong request configuration');
        }
    }

    /**
     * @return array
     */
    public function exportToArray(): array
    {
        return $this->temperaturesRepository->findTemperaturesByConditions($this->conditions, $this->inConditions);
    }

    /**
     * @return string
     */
    public function exportToExcel(): string
    {
        $temperaturesArr = $this->exportToArray();
        $temperaturesArr = $this->prepareTemperaturesArray($temperaturesArr);
        $exportHelper = new ExcelExportService($this->excelHeaders, $temperaturesArr, time().'_exported', $this->columnsAllowedToMerge);
        $exportHelper->write();

        return '';
    }

    /**
     * @param string $date
     * @return bool
     */
    private function validateDate(string $date)
    {
        return preg_match('/\d{4}-\d{2}-\d{2}/', $date) === 1;
    }
```

```
    /**
     * @param array $temperaturesArr
     * @return array
     */
    private function prepareTemperaturesArray(array $temperaturesArr): array
    {
        $result = [];
        foreach ($temperaturesArr as $i => $arr) {
            preg_match('/\d{4}-\d{2}-\d{2}/', $arr['updated_at'], $matches);
            $arr['date'] = $matches[0];
            preg_match('/\d{2}:\d{2}:\d{2}/', $arr['updated_at'], $matches);
            $arr['time'] = $matches[0];

            foreach ($arr as $key => $value) {
                if(in_array($key, array_keys($this->excelHeaders))) {
                    $result[$i][$key] = $value;
                }
            }
            $i++;
        }

        return $result;
    }
}


//клас, що перевіряє рівень доступу користувача
class RoleChecker
{

    public function check(AccessibleInterface $user, string $role): bool
    {
        $result = DB::selectOne("select CASE
```

```
        WHEN (select level FROM user_roles WHERE title = ?) > (select level FROM user_roles WHERE title
= ?) THEN true
        WHEN (select level FROM user_roles WHERE title = ?) = (select level FROM user_roles WHERE title
= ?)
            AND ? = ? THEN true
        ELSE false
        END as 'result';", [$user->getRole(), $role, $user->getRole(), $role, $user->getRole(), $role]);

      return (bool)$result->result;
   }
}
```

//контролер модулю температур

```
<?php

namespace App\Http\Controllers;

use App\Repositories\Interfaces\TemperaturesRepositoryInterface;
use App\TemperaturesExportService\TemperaturesExportService;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;

class TemperatureController extends Controller
{
   /** @var TemperaturesRepositoryInterface */
   private $temperaturesRepository;

   /** @var TemperaturesExportService */
   private $temperaturesExportService;

   public function __construct(TemperaturesRepositoryInterface $repository)
   {
```

```
    $this->temperaturesRepository = $repository;
    $this->temperaturesExportService = new TemperaturesExportService($repository);
}

public function addTemperature(Request $request)
{
    try {
        $requestData = $request->json()->all();
        $this->temperaturesRepository->addTemperature($requestData);

        return response()->json(['status' => 'OK']);
    } catch (\Throwable $exception) {
        return response()->json(['status' => 'ERROR']);
    }
}

/**
 * @param Request $request
 * Приймає на вхід наступні параметри:
 * date - якщо одне значення - пошук на конкретну дату. Якщо масив - має містити ключі from і to
 * room - пошук у конкретній кімнаті
 * temperatures
 * @return JsonResponse
 */
public function getTemperatures(Request $request): JsonResponse
{
    $requestData = $request->json()->all();

    try {
        $this->temperaturesExportService->configure($requestData);
        $temperatures = $this->temperaturesExportService->exportToArray();

        return response()->json(['status' => 'OK', 'data' => $temperatures]);
    } catch (\Throwable $exception) {
        return response()->json(['status' => 'ERROR', 'message' => $exception->getMessage()]);
```

```
        }
    }

    public function exportTemperatures(Request $request)
    {
        $requestData = $request->json()->all();

        try {
            $this->temperaturesExportService->configure($requestData);
            $link = $this->temperaturesExportService->exportToExcel();

            return response()->json(['status' => 'OK', 'data' => $link]);
        } catch (\Throwable $e) {
            return response()->json(['status' => 'ERROR', 'message' => $e->getMessage()]);

        }
    }
}

//компонент для перевірки ролі користувача
class CheckUserRole
{

    protected $roleChecker;

    public function __construct(RoleChecker $roleChecker)
    {
        $this->roleChecker = $roleChecker;
    }

    /**
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure $next
```

```
     * @param $role
     * @return mixed
     */
    public function handle($request, Closure $next, $role)
    {
        /** @var User $user */
        $user = Auth::guard('api')->user();
        if(!$this->roleChecker->check($user, $role)) {
            throw new AuthenticationException('Access Forbidden');
        }

        return $next($request);
    }
}

//контролер що відповідає за інформацію про будівлі

class BuildingsController
{
    /** @var BuildingsRepositoryInterface */
    private $buildingsRepository;

    public function __construct(BuildingsRepositoryInterface $buildingsRepository)
    {
        $this->buildingsRepository = $buildingsRepository;
    }

    public function getBuildingInfo(int $id): Response
    {
        $data = $this->buildingsRepository->findById($id);

        return response()->json($data->toArray());
    }

    public function getAllBuildingsInfo(): Response
```

```
{
    $buildingsData = $this->buildingsRepository->findAll();

    return response()->json($buildingsData);
}

public function getAllBuildingsTitles(): Response
{
    $titles = array_map(function (array $buildingData) {
        return [
            'id' => $buildingData['id'],
            'title' => $buildingData['title']
        ];
    }, $this->buildingsRepository->findAll());

    return response()->json($titles);
}

public function deleteBuildingInfo(int $id): Response
{
    $deleted = $this->buildingsRepository->deleteBuildingData($id);
    $result = $deleted ? ['message' => 'Deleted successfully'] : ['error' => 'Something went wrong'];

    return response()->json($result);
}

public function addBuilding(Request $request)
{
    try {
        $requestData = $request->json()->all();
        $this->buildingsRepository->addBuilding($requestData);

        return response()->json(['status' => 'OK']);
    } catch (\Throwable $exception) {
        return response()->json(['status' => 'ERROR']);
```

```
    }
}

public function editBuilding(Request $request)
{
    try {
        $requestData = $request->json()->all();
        if(!isset($requestData['id'])) {
            return response()->json(['status' => 'ERROR', 'message' => 'Required param ID mised']);
        }
        $this->buildingsRepository->editBuilding($requestData);

        return response()->json(['status' => 'OK']);
    } catch (\Throwable $exception) {
        return response()->json(['status' => 'ERROR']);
    }
}

public function getRelatedRooms(int $id)
{
    $rooms = $this->buildingsRepository->getRelatedRooms($id);

    return response()->json(['status' => 'OK', 'data' => $rooms]);
}
```

Додаток В

Система збору даних про енергоресурси та температурні режими для служби енергоменеджменту

Опис програми

УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ6122_20Б

Аркушів 1

Київ — 2019

АНОТАЦІЯ

Додаток надає інформацію про функціонування модулів перевірки ролі користувача, та експорту даних про температури в аудиторіях у звіт в форматі Excel.

Розроблена система дає можливість зберігати, обробляти та отримувати доступ до даних про будівлі, аудиторії, температури та показники теплових лічильників у них, із розділенням за ролями користувачів. Доступна також можливість експорту даних по температурам у формат Excel.

Систему було розроблено за допомогою мови програмування PHP, фреймворку Laravel, СКБД MySQL та бібліотеки PhpSpreadsheet.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

Додаток розроблено на мові програмування PHP 7.3 із використанням фреймворку Laravel, СКБД MySQL та бібліотеки PhpSpreadsheet. Для функціонування додатку необхідно використовувати будь-який із сучасних веб-серверів, що підтримують роботу із PHP (Apache, nginx, тощо).

Після запуску сервера із програмним забезпеченням, додаток готовий приймати та обробляти запити від третіх сторін (клієнтська частина додатку або ж стороннє API, що надсилає данні).

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Система має два основні модулі – модуль загальної інформації, та модуль тепломоніторингу. У окремий модуль також можна винести підсистему авторизації та перевірки ролі та рівня доступу користувача.

Розроблені модулі призначені автоматизувати роботу служби енергоменеджменту.

Модуль загальної інформації включає в себе наступний функціонал:

- Перегляд інформації про корпус
- Додавання інформації про корпус
- Редагування інформації про корпус
- Перегляд аудиторій корпусу
- Перегляд інформації про аудиторію
- Редагування інформації про аудиторію
- Додавання інформації про аудиторію.

Модуль тепломоніторингу включає в себе наступний функціонал:

- Додавання лічильника
- Редагування лічильника
- Внесення показників лічильника
- Додавання інформації про температури
- Перегляд інформації про температури
- Експорт даних про температури у форматі Excel

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Система складається із незалежних один від одного компонентів, що призначені вирішувати поставлені завдання. При розробці системи враховувались сучасні підходи до побудови архітектури програмного забезпечення, таким чином класи у системі не залежать від конкретних реалізацій та об’єктів, а від абстракцій (інтерфейсів).

Прийом запитів виконують контролери, що передають керування у сервіси або отримують інформацію із репозиторіїв у випадку простих дій. Репозиторії відповідають за роботу із базою.

Таким чином, система може бути легко масштабована у майбутньому.

ТЕХНЧНІ ЗАСОБИ, ЩО ВИКОРИСТОВУЮТЬСЯ

Систему розроблено за допомогою наступних засобів: IDE PHPStorm, сервер Apache, система контролю версій Git та програма для тестування API –Postman.

Для написання системи було обрано мову програмування PHP, СКБД MySQL, фреймворк Laravel та бібліотеку для роботи із Excel – Phpspreadsheet.

ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідними даними системи є запити із клієнтської частини або стороннього API. Запити складаються із заголовків та тіла, відповідно до цих параметрів ідентифікується контролер та метод що повинен обробляти запит.

У якості вихідних даних система повертає відповідь у форматі JSON, що містить статус та дані.